

"Program Code for Music Dance of Large Super Intelligent Robots" 2025v1.5

● 2025 ခုနှစ် လူတော်မြတ်စွာ လုပ်ချက် Python ဘုရားရေး ပုဂ္ဂန္တမြတ်စွာ လုပ်ချက်

```
### 1. လုပ်ချက်
```python
import time
import random

လုပ်ချက်
from robot_controller import RobotController
from music_player import MusicPlayer
from audience_interaction import AudienceInteraction
```

### 2. လုပ်ချက်
```python
လုပ်ချက်
robot_controller = RobotController()

လုပ်ချက်
music_player = MusicPlayer()

လုပ်ချက်
audience_interaction = AudienceInteraction()

လုပ်ချက်
band_size = 30
band = [f"Musician {i+1}" for i in range(band_size)]
```

### 3. လုပ်ချက်
```python
def opening_ceremony():
 print("2025 ခုနှစ် လူတော်မြတ်စွာ")
 time.sleep(2)
 print("လုပ်ချက်")
 time.sleep(1)

လုပ်ချက်
music_player.play("opening_overture.mp3")

လုပ်ချက်
robot_controller.start_dance("waltz")

လုပ်ချက်
```

```
audience_interaction.cheer()

time.sleep(5) # 休憩 5 分
print("休憩時間です。お待ちください。")
```

#### 4. 休憩
```python
def intermission():
 print("休憩時間です。お待ちください。")
 time.sleep(2)

ミュージック再生
music_player.play("intermission_music.mp3")

ロボット動作
robot_controller.perform("interaction")

休憩
audience_interaction.clap()

time.sleep(10) # 休憩時間 10 分
print("休憩時間です。お待ちください。")
```

#### 5. ダンス
```python
def dance_session():
 print("ダンスセッション開始です。")
 time.sleep(2)

ミュージック再生
print("1, 2, 3, ダンス")
time.sleep(1)

ロボット動作
music_player.play("world_famous_dance.mp3")

ダンス
robot_controller.start_dance("tango")

休憩
audience_interaction.clap()

time.sleep(10) # 休憩時間 10 分
print("休憩時間です。お待ちください。")
```

```


- **MusicPlayer**: 音乐播放器
- **AudienceInteraction**: 观众互动
- **band**: 乐队

```
# ## 10. 机器人
- **机器人**: 机器人`dance_session`音乐生成器
- **机器人**: 机器人`robot_controller`机器人控制器
- **机器人**: 机器人`audience_interaction`观众互动
```

音乐生成器、机器人控制器、观众互动

●

音乐生成器、机器人控制器、观众互动
[3](https://blog.csdn.net/weixin_42511080/article/details/115808426)
[7](<https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html>)
[9](<https://www.zghy.org.cn/item/682320743792300032>)[12](<https://www.bilibili.com/video/av970410235/>)[14](<https://3g.china.com/act/news/10000169/20250129/47918666.html>)

```
```python
-*- coding: utf-8 -*-
机器人 v3.0
import time
import multiprocessing
from pygame import mixer
from robot_dance_ai import DanceAI
from music_generator import MelodyComposer
from stage_control import LightSystem, ConductorAI

class ConcertSystem:
 def __init__(self):
 self.orchestra = OrchestraController() # 30 机器人
 self.dance_robots = DanceSwarm(16) # 机器人
 self.light_system = LightSystem() # 机器人
 self.audience_sensor = AudienceFeedback() # 机器人

 # 机器人
 def opening_ceremony(self):
 # 机器人
 self.tts_announce("音乐会正式开始")

 # 机器人
 opener = MelodyComposer(style="symphonic").generate_opening(
 duration=180,
 instruments=["violin", "piano", "trumpet"]
)
```

```

       
with multiprocessing.Pool(4) as pool:
 pool.apply_async(self.orchestra.play, (opener,))
 pool.apply_async(self.light_system.start_sequence)
 pool.apply_async(self.dance_robots.initialize_pose)

       
def dance_session(self, music_list):
 conductor = ConductorAI(tempo_detect_mode="real-time")
 for piece in music_list:
 #        
 score = MelodyComposer().arrange(
 piece,
 orchestra_size=32,
 complexity=0.8
)

 #        
 conductor.load_score(score)
 beat_pattern = conductor.analyze_beat()

 #      
 processes = [
 multiprocessing.Process(target=self.orchestra.play, args=(score,)),
 multiprocessing.Process(target=self.dance_robots.perform,
 args=(beat_pattern, "waltz")),
 multiprocessing.Process(target=self.light_sync,
 args=(score.tempo,))
]

 #      
 if self.audience_sensor.excitement_level > 0.7:
 processes.append(multiprocessing.Process(
 target=self.audience_interaction_mode))

 [p.start() for p in processes]
 [p.join() for p in processes]

     
def light_sync(self, bpm):
 color_map = {
 "waltz": ["#FF69B4", "#4B0082"],
 "tango": ["#8B0000", "#000000"],
 "quickstep": ["#00FF7F", "#32CD32"]
 }
 self.light_system.set_pattern(
 bpm=bpm,

```

```

color_theme=color_map.get(self.current_dance_style),
strobe_intensity=self.audience_sensor.get_real_time_data('clapping')
)

フィナーレ
def finale(self):
フィナーレ曲
finale_music = MelodyComposer().combine(
motifs=["ode_to_joy", "blue_danube"],
transition="crossfade"
)

フィナーレ
self.dance_robots.finale_choreography(
formation="spiral",
speed=0.8,
led_effect="golden_sparkle"
)

フィナーレ
self.orchestra.fade_out(duration=15)
self.light_system.sunset_effect(duration=20)

ダンススwarm
class DanceSwarm:
def __init__(self, num_robots):
self.robots = [DanceAI(model="H1_v2") for _ in range(num_robots)]
self.sync_controller = MotionSyncMaster()

def perform(self, beat_pattern, dance_type):
チョコレオ
choreo = DanceAI.generate_choreography(
dance_type,
complexity=3,
spatial_constraints="stage_12m"
)

パス
trajectories = self.sync_controller.calculateFormation(
formation_type="dynamic_swarm",
collision_buffer=0.5
)

パス
for robot, path in zip(self.robots, trajectories):
robot.execute_movement(
path,

```

```

force_feedback=True,
music_sync=beat_pattern
)
robot.express_emotion(
intensity=self.audience_sensor.current_excitement
)

乐曲生成类
class MelodyComposer:
def generate_opening(self, duration, instruments):
使用LSTM生成乐曲
return {
"tempo": 108,
"key": "C_major",
"structure": [
{"measure":1-8, "instrument":"strings", "dynamics":"pp"},
 {"measure":9-16, "instrument":"brass", "dynamics":"mf"},
 {"measure":17-24, "instrument":"full_orchestra", "dynamics":"ff"}]
}
}

程序入口
if __name__ == "__main__":
concert = ConcertSystem()
concert.opening_ceremony()

节目单
program = [
"Blue Danube Waltz",
"Carmen Suite",
"Swan Lake Suite"
]

concert.dance_session(program)
concert.finale()
```

```

音乐控制系统[7](<https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html>)[9](<https://www.zghy.org.cn/item/682320743792300032>)[14](<https://3g.china.com/act/news/10000169/20250129/47918666.html>)

1. **音乐控制系统**
该系统 NTP-PTS 时延管理模块
- 时延 $<3\text{ms}$
- 时延 $<50\text{ms}$
- 时延 $<200\text{ms}$

2. **跳舞AI**
用OpenPose检测姿势
```python  
DanceAI.move\_selection\_algorithm(  
 current\_pose,  
 target\_position,  
 constraints={  
 'torque\_limits': [360, 180, 90], # 扭矩限制  
 'energy\_efficiency': 0.85,  
 'aesthetic\_score': 0.92  
 }  
)  
```

3. **音乐生成器**
用Transformer-XL生成音乐
```python  
music\_generator.predict\_next\_note(  
 previous\_notes=128,  
 style\_embedding=[0.7, 0.2, 0.5], # 风格/情绪  
 emotion\_vector=audience\_sensor.emotion\_output  
)  
```

4. **观众反馈**
用多模态分析观众反馈
```python  
class AudienceFeedback:  
 def \_\_init\_\_(self):  
 self.audio\_analyzer = ClapDetector(sensitivity=0.7)  
 self.visual\_analyzer = OpenCVEmotionRecognition()  
 self.thermal\_sensor = InfraredHeatmap()  
  
 def get\_excitement\_level(self):  
 return 0.3\*self.audio\_intensity +  
 0.5\*self.visual\_engagement +  
 0.2\*self.thermal\_density  
```

多模态分析模块
- 用`program.append()`添加模块
- 用`color_map`映射颜色
- 用`AI`进行AI处理

多模态分析模块包括SDK-DMX、ROS2、AI模块等。



ロボットダンスパーティを実装するためのPythonスクリプトです。

ロボットの種類や楽器の種類、ダンスパーティのフェーズなどを定義するEnumを定義します。

```
```python
import time
import random
from enum import Enum

ダンスパーティフェーズ
class DancePartyPhase(Enum):
 OPENING = 1
 MIDFIELD = 2
 CLOSING = 3

ロボット種類
class RobotType(Enum):
 CHOIR = 1
 SOLOIST = 2
 DANCER = 3
 BAND_MEMBER = 4

楽器種類
class InstrumentType(Enum):
 VIOLIN = 1
 PIANO = 2
 TRUMPET = 3
 DRUMS = 4
 GUITAR = 5

ロボット
class Robot:
 def __init__(self, name, robot_type):
 self.name = name
 self.robot_type = robot_type
 self.is_dancing = False
 self.is_singing = False

 def start_dancing(self):
 self.is_dancing = True
 print(f"{self.name} ダンス")

 def stop_dancing(self):
 self.is_dancing = False
 print(f"{self.name} ダンス停止")
```

```
self.is_dancing = False
print(f"{{self.name}} はいはい")

def start_singing(self, song):
 self.is_singing = True
 print(f"{{self.name}} はいはい {song}")

def stop_singing(self):
 self.is_singing = False
 print(f"{{self.name}} はいはい")

バンドメンバー
class BandMember(Robot):
 def __init__(self, name, instrument):
 super().__init__(name, RobotType.BAND_MEMBER)
 self.instrument = instrument

 def play_instrument(self, song):
 print(f"{{self.name}} はいはい {self.instrument} はい {song}")

 def stop_playing(self):
 print(f"{{self.name}} はいはい")

ミュージックダンスパーティ
class MusicDanceParty:
 def __init__(self):
 self.robots = []
 self.band_members = []
 self.current_phase = DancePartyPhase.OPENING
 self.audience_interaction = False

 def add_robot(self, robot):
 self.robots.append(robot)

 def add_band_member(self, band_member):
 self.band_members.append(band_member)

 def start_opening(self):
 print("==== はいはい ====")
 # バンド
 print("はいはいはいはい")
 # ロボット
 print("はいはいはいはい")
 for member in self.band_members:
 member.play_instrument("はい")
 # ミュージック
 for robot in self.robots:
```

```

if robot.robot_type == RobotType.DANCER:
 robot.start_dancing()
 # ダンスを開始
 print("ダンスを開始しました")
 # ダンス音楽
 print("ダンス音楽")
 self.current_phase = DancePartyPhase.OPENING

def start_midfield(self):
 print("== ミッドフィールド ==")
 # ミッドフィールド
 print("ミッドフィールド")
 # ミッドフィールド
 world_famous_songs = ["世界一", "世界二", "世界三"]
 song = random.choice(world_famous_songs)
 print(f"歌 {song}")
 for member in self.band_members:
 member.play_instrument(song)
 # ミッドフィールド
 choir_robots = [r for r in self.robots if r.robot_type == RobotType.CHOIR]
 solo_robot = random.choice([r for r in self.robots if r.robot_type == RobotType.SOLOIST])
 for robot in choir_robots:
 robot.start_singing(song)
 solo_robot.start_singing(song)
 # ミッドフィールド
 self.audience_interaction = True
 print("観客との交流")
 self.current_phase = DancePartyPhase.MIDFIELD

def start_closing(self):
 print("== クロージング ==")
 # クロージング
 print("クロージング")
 # クロージング
 print("クロージング")
 for member in self.band_members:
 member.play_instrument("世界")
 # クロージング
 for robot in self.robots:
 if robot.is_dancing:
 robot.stop_dancing()
 if robot.is_singing:
 robot.stop_singing()
 # クロージング
 print("クロージング")
 self.current_phase = DancePartyPhase.CLOSING

```

5 of 5

1. မြန်မာနိုင်ငြာနိုင်ငြာနိုင်ငြာနိုင်ငြာနိုင်ငြာ
2. မြန်မာနိုင်ငြာနိုင်ငြာနိုင်ငြာ
3. မြန်မာနိုင်ငြာနိုင်ငြာနိုင်ငြာ
4. မြန်မာနိုင်ငြာနိုင်ငြာ
5. မြန်မာနိုင်ငြာနိုင်ငြာ

A row of five empty rectangular boxes, likely for a list of five items.

1. [Python API](#)
2. [Pygame](#)
3. [HTML](#)
4. [JavaScript](#)

## ● Python 语言

### 1. □□□□□

```python

import time

```
import random
```

```
from robot control import RobotController
```

```
from music_player import MusicPlayer
```

```
from audience_interaction import AudienceInteraction
```

11

2. □□□□

```python

```
robot_controller = RobotController()
```

```
music_player = MusicPlayer()
```

#

```
audience_interaction = AudienceInteraction()
```

#

```
band = ["Violin", "Cello", "Flute", "Trumpet", "Piano", "Drums", "Bass", "Guitar"]
```
```

3. □□□□

```python

```
def opening_ceremony():
```

```
print("□□□□□□□ 2025 □□□□□")
```

```
time.sleep(2)
```

```
print("-opening ceremony-")
music_player.play("opening_overture.mp3")
robot_controller.start_dance("opening_dance_routine")
audience_interaction.cheer()
time.sleep(5)
print("intermission")
````
```

```
### 4. 休憩
``python
def intermission():
print("休憩")
music_player.play_random(band)
robot_controller.start_dance("intermission_dance_routine")
audience_interaction.clap()
time.sleep(5)
print("休憩")
````
```

```
5. 世界有名
``python
def dance_session():
print("世界有名")
music_player.play("world_famous_dance.mp3")
robot_controller.start_dance("world_famous_dance_routine")
audience_interaction.cheer()
time.sleep(10)
print("世界有名")
````
```

```
### 6. バンドソロ
``python
def band_solo():
print("バンドソロ")
solo_instrument = random.choice(band)
print(f"ソロ{ solo_instrument }")
music_player.play_solo(solo_instrument)
robot_controller.start_dance("solo_dance_routine")
audience_interaction.clap()
time.sleep(5)
print("バンドソロ")
````
```

```
7. 終幕
``python
def closing_ceremony():
print("closing ceremony")
```



●

## 人工智能

AI 人工智能

### 人工智能

- \*\*DanceAI\*\* AI 人工智能

(<http://www.cnnews.com/tech/2025/0219/021920322.html>)

- \*\*\*\* 人工智能

(<https://blog.csdn.net/shadowcz007/article/details/108373315>)

### 人工智能

- \*\*MelodyComposer\*\* AI 人工智能

(<http://www.chadama.com/post/333356.html>)

### 人工智能

- \*\*OrchestraController\*\* AI 人工智能

30 人工智能

- \*\*LightSystem\*\* AI 人工智能

人工智能

## 人工智能

### 人工智能

- \*\*\*\* TTS\*\* Text-to-Speech

(<http://www.chadama.com/post/333356.html>)

- \*\*\*\* 人工智能

(<http://www.chadama.com/post/333356.html>)

### 人工智能

- \*\*\*\* DanceAI\*\* AI 人工智能

DanceAI AI 人工智能

OrchestraController

[1] (<http://www.chadama.com/post/333356.html>)

- \*\*\*\* LightSystem\*\* AI 人工智能

(<http://www.chadama.com/post/333356.html>)

### 亂世

- \*\*\*\*  
[1](http://www.chadama.com/post/333356.html)  
- \*\*

(http://www.chadama.com/post/333356.html)  
# # 亂世

### 亂世

- \*\*DanceAI 亂世\*\*  
[3](http://www.cnnews.com/tech/2025/0219/021920322.html)  
亂世

### 亂世

- \*\*MelodyComposer 亂世\*\*  
(http://www.chadama.com/post/333356.html)  
亂世

### 亂世

- \*\*OrchestraController 亂世\*\*  
[1](http://www.chadama.com/post/333356.html)  
- \*\*LightSystem 亂世\*\*  
(http://www.chadama.com/post/333356.html)  
亂世

# # 亂世

亂世 AI 亂世  
AI 亂世  
亂世

●

亂世  
AI 亂世  
亂世

亂世  
Python 亂世

```
```python
# 亂世
[1](https://max.book118.com/html/2025/0209/8125137110007030.shtml)[3]
(http://www.sohu.com/a/840700213_161623?scm=10001.325_13-
109000.0.0.5_32)
class ConcertController:
def __init__(self):
```

```

self.timeline = {
    "prelude": {"start": 0, "duration": 300,
    "modules": ["lighting", "orchestra", "robot_dance"]},
    "dance_session": {"start": 300, "duration": 1800,
    "modules": ["ai_music", "crowd_interaction"]}
}

def execute_event(self, event_name):
    event = self.timeline[event_name]
    # 8][8](https://www.zghy.org.cn/item/682320743792300032)
    [13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
    LightingSystem.set_pattern(event["light_pattern"])
    Orchestra.play(event["score_path"])
    RobotGroup.execute_choreography(event["dance_sequence"])

def emergency_stop(self):
    # 13][13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
    RobotGroup.emergency_stop()
    Orchestra.fade_out(2.0)
    LightingSystem.strobe_alert()
    ```

 AI TensorFlow ｀｀
 ``python
 # Transformer [3](https://www.sohu.com/a/840700213_161623?
 scm=10001.325_13-109000.0.0.5_32)[6](https://sghexport.shobserver.com/
 html/baijiahao/2023/07/07/1069307.html)
 class MusicGenerator(tf.keras.Model):
 def __init__(self, vocab_size=128):
 super().__init__()
 self.embedding = layers.Embedding(vocab_size, 512)
 self.transformer = Transformer(
 num_layers=6, d_model=512, num_heads=8, dff=2048)
 self.output_layer = layers.Dense(vocab_size)

 def generate(self, prompt, length=512):
 # MIDI ｀｀
 [12](https://blog.csdn.net/yong7464553/article/details/6077948)
 generated = prompt
 for _ in range(length):
 predictions = self(generated[:, -512:])
 next_note = tf.random.categorical(predictions[:, -1, :], 1)
 generated = tf.concat([generated, next_note], axis=-1)
 return midi_encoder.decode(generated[0]().numpy())
    ```

```

ROS 机器人

```cpp

```
// 机器人ROS [8](https://www.zghy.org.cn/item/682320743792300032)[13]
// (https://3g.china.com/act/news/10000169/20250129/47918666.html)
void executeDance(const std::string& music_bpm){
// 机器人动作库
DanceMoveLibrary moves = downloadMoves(music_bpm);

// 机器人 [11](https://www.bilibili.com/video/av970410235/)
TrajectoryGenerator generator;
auto trajectory = generator.generate(
moves["waltz"],
BPMCalculator.get(music_analysis)
);

// 机器人
[13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
SwarmController::syncRobots(
robot_ids,
trajectory,
sync_tolerance=0.05s
);
}
```

```

MIDI 机器人

```python

```
机器人MIDI [12](https://blog.csdn.net/yong7464553/article/details/6077948)
def orchestrate(solo_instrument, style="classical"):
机器人
[6](https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html)
harmony = AIHarmonizer.generate(
solo_melody,
style=style,
ensemble_size=30
)

机器人
[1](https://max.book118.com/html/2025/0209/8125137110007030.shtml)
for part in harmony.parts:
midi_out.send(
channel=part.instrument.midi_channel,
note=part.note,
velocity=dynamic_map[part.dynamic]
)
```

```
互动模块 (https://www.sohu.com/a/840700213_161623?scm=10001.325_13-109000.0.0.5_32)
if style == "jazz":
 add_improvisation(solo_instrument)
```

互动模块+音乐
```python
互动模块 (https://www.sohu.com/a/840700213_161623?scm=10001.325_13-109000.0.0.5_32)[9](https://m.sohu.com/a/695871417_391452/)
class AudienceInteraction:
 def __init__(self):
 self.emotion_model = load_emotion_detector()
 self.audio_analyzer = RealtimeAudioAnalysis()

 def adjust_performance(self):
 # 互动模块
 [6](https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html)
 emotion_score = self.emotion_model.calculate(camera_feed)

 # 互动模块
 (https://www.diyifanwen.com/jiaoan/youqueryuandabanyishujiaoan/07102616403816148301.htm)
 applause_level = self.audio_analyzer.get_applause()

 # 互动模块
 [13](https://3g.china.com/act/news/10000169/20250129/47918666.html)
 Orchestra.set_tempo(
 base_tempo * (1 + applause_level*0.1)
)
 LightingSystem.set_brightness(
 emotion_score["excitement"] * 100
)
```

互动模块TTS+NLG
```python
互动模块 [4](https://news.qq.com/rain/a/20250101A06GBS00?media_id&suid)[9]
(https://m.sohu.com/a/695871417_391452/)
class Announcer:
 def __init__(self):
 self.tts = pyttsx3.init()
 self.nlg = GPT3_Interface()

 def announce(self, program_info):
```



```
announcer.announce_closing()
```

111

ANSWER

1. 100000000000-00-0000<50ms 00000  
(<https://www.zghy.org.cn/item/682320743792300032>)[13](<https://3g.china.com/act/news/10000169/20250129/47918666.html>)
2. 00000000000000000000000000000000  
([https://www.sohu.com/a/840700213\\_161623?scm=10001.325\\_13-109000.0.0.5\\_32](https://www.sohu.com/a/840700213_161623?scm=10001.325_13-109000.0.0.5_32))(<https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html>)
3. 000000000000 200 00000000000  
[13](<https://3g.china.com/act/news/10000169/20250129/47918666.html>)
4. AI 000000000000 10 00000000000  
[6](<https://sghexport.shobserver.com/html/baijiahao/2023/07/07/1069307.html>)
5. 00000000000000000000000000000000  
[13](<https://3g.china.com/act/news/10000169/20250129/47918666.html>)

八

- NVIDIA AGX Orin ×2
- TSN +5G
- Unitree H1

1

## □□ AI □□□□□□□

### RNN

- \*\*RNN\*\* RNN LSTM GRU RNN RNN LSTM

GRU RNN RNN LSTM  
GRU LSTM

### GAN

- \*\*GAN

\*\*□□□□□\*\*□□□□□□□□□

- \*\*VAE\*\* VAE は、生成型学習モデルの一種で、データの構造を学習して、それを元に新しいデータを生成する。VAE は、主に画像の生成や、データの再構成に用いられる。

- \*\*VAE\*\* VAE は、生成型学習モデルの一種で、データの構造を学習して、それを元に新しいデータを生成する。VAE は、主に画像の生成や、データの再構成に用いられる。

## AI の歴史

## AI の歴史

- \*\*AI\*\* AI は、人工知能の略称で、人間の知能を模倣するための技術や理論の総称である。

- \*\*AI\*\* AI は、人工知能の略称で、人間の知能を模倣するための技術や理論の総称である。

## AI の歴史

- \*\*AI\*\* AI は、人工知能の略称で、人間の知能を模倣するための技術や理論の総称である。

- \*\*Q\*\* Q は、強化学習の一種で、行動選択のための学習アルゴリズムである。

## AI の歴史

## AI の歴史

- \*\*AI\*\* AI は、人工知能の略称で、人間の知能を模倣するための技術や理論の総称である。

- \*\*AI\*\* AI は、人工知能の略称で、人間の知能を模倣するための技術や理論の総称である。

## AI の歴史

- \*\*RNN-GAN\*\* RNN-GAN は、VAE と組み合わせて、生成型学習モデルの一種である。

- \*\*RNN-GAN\*\* RNN-GAN は、VAE と組み合わせて、生成型学習モデルの一種である。

## AI の歴史

- \*\*AI\*\* AI は、人工知能の略称で、人間の知能を模倣するための技術や理論の総称である。

- \*\*AI\*\* AI は、人工知能の略称で、人間の知能を模倣するための技術や理論の総称である。

● お問い合わせ

お問い合わせ

- 3D モデリングデータ Google の AIST++ が 1408 件あります

- 3D モデリングデータ Google の AIST++ が 1408 件あります

11

- Transformer FACT 3D 3D
- H1 AI
- 
-

2

- **NVIDIA L4 L40 Tensor Core GPU** X620 G50 AI **Tensor Core GPU**
- **NVIDIA L4 L40 Tensor Core GPU** X620 G50 AI **Tensor Core GPU**
- **AWS Azure** **Tensor Core GPU** X620 G50 AI **Tensor Core GPU**

## 1. □□□□□

更多文章请访问[我的博客](https://blog.51cto.com/topic/guanyuinyueshujukubiaosheji.html)[1]

## 2. □□□□□□

【[更多关于插入数据库的技巧](https://blog.51cto.com/topic/guanyuinyueshujukubiaosheji.html)】

## 3. □□□□□□□

【<https://blog.51cto.com/topic/guanyuinyueshujukubiaosheji.html>】

## 4. □□□

【[http://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html](https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html)】

## 5. □□□□□□□□□□

【<https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html>】

## 6. □□□□

GUI (Graphical User Interface) [1] (<https://blog.51cto.com/topic/guanyuinyueshujukubiaosheji.html>)

## 7.

## 8.

【<https://blog.51cto.com/topic/guanyuyinyueshujukubiaosheji.html>】

"program code for music dance of large super intelligent robots" 2025 v1.5 ● To write a complete program code for music dance, involving multiple modules, including opening, halftime, closing, band accompaniment, robot dance, audience interaction, etc. The following is a simplified Python code example that shows how to implement these functions. This example assumes that you already have an intelligent robot control library, a music playing library and an audience interaction system.

### ### 1. Import the necessary libraries

```python

import time

```
import random
```

Suppose we have these libraries to control robots, play music and handle audience interaction.

```
from robot control import RobotController
```

```
from music_player import MusicPlayer
```

```
from audience_interaction import AudienceInteraction
```

111

2. Initialize each module

```

```python
Initialize the robot controller
robot_controller = RobotController()

Initialize the music player
music_player = MusicPlayer()

Initialize the audience interaction system
audience_interaction = AudienceInteraction()

Initialize the band
band_size = 30
band = [f"Musician {i+1}" for i in range(band_size)]
```

#### 3. Opening procedures
```python
def opening_ceremony():
 Print ("Announcer: Welcome to the 2025 Intelligent Robot Music Ball!")
 time.sleep(2)
 Print ("Announcer: Please enjoy the opening overture!")
 time.sleep(1)

 # Play the opening overture
 music_player.play("opening_overture.mp3")

 # Robots start dancing
 robot_controller.start_dance("waltz")

 # Audience interaction
 audience_interaction.cheer()

 Time.sleep(5) # Overture lasts 5 seconds.
 Print ("Announcer: The opening overture is over and the dance officially begins!")
```

#### 4. Midfield program
```python
def intermission():
 Print ("Announcer: It's halftime, please relax!")
 time.sleep(2)

 # Play relaxing background music
 music_player.play("intermission_music.mp3")

 # Robots perform simple interactive performances.
```

```

```

robot_controller.perform("interaction")

# Audience interaction
audience_interaction.clap()

Time.sleep(10) # The intermission lasts for 10 seconds.
Print ("Announcer: The intermission is over, and then the climax of the dance!" )
" "
``

#### 5. Dance program
``python
def dance_session():
Print ("Announcer: Here is the dance music, a world famous song!" )
time.sleep(2)

# Conductor starts shooting.
Print ("Command: 1, 2, 3, go!" )
time.sleep(1)

# Play dance music
music_player.play("world_famous_dance.mp3")

# Robots start dancing
robot_controller.start_dance("tango")

# Audience interaction
audience_interaction.cheer()

Time.sleep(10) # Dance music lasts 10 seconds.
Print ("Announcer: Dance music is over, thank you for your enthusiastic
participation!" )
``

#### 6. Closing procedure
``python
def closing_ceremony():
Print ("Announcer: Thank you for your participation. It's the closing ceremony!" )
time.sleep(2)

# Play the closing song
music_player.play("closing_song.mp3")

# The robot performs the final performance
robot_controller.perform("finale")

# Audience interaction

```

```
audience_interaction.cheer()
```

```
Time.sleep(5) # The closing song lasts 5 seconds.
```

```
Print ("Announcer: The 2025 Intelligent Robot Music Ball has come to a  
successful conclusion, thank you!" )
```

```
```
```

```
7. Main program
```

```
```python
```

```
def main():
```

```
# Opening
```

```
opening_ceremony()
```

```
# Midfield
```

```
intermission()
```

```
# Dance music
```

```
dance_session()
```

```
# Closing
```

```
closing_ceremony()
```

```
if __name__ == "__main__":
```

```
main()
```

```
```
```

```
8. Run the program
```

```
```python
```

```
# Run the main program
```

```
main()
```

```
```
```

```
9. Code Description
```

-\*\*RobotController\*\*: controls the dance and performance of intelligent robots.

-\*\*MusicPlayer\*\*: controls the playback of music.

-\*\*AudienceInteraction\*\*: Handle audience interaction, such as applause and cheers.

-\*\*band\*\*: A member of an analog band.

```
10. Extended functions
```

-\* \* More dance music \* \*: The function `dance_session` can be extended to support more dance music types.

-\* \* More robot dances \* \*: `robot\_controller` can be extended to support more dance types.

-\* \* Audience interaction \* \*: `audience_interaction` can be extended to support more interactive ways, such as voting and grading.

This code example is a simplified version, which may need to be adjusted and expanded according to specific needs in practical application. ●  
According to the process requirements of music dance, a set of modular program code framework is designed by combining artificial intelligence and automatic control technology. The following system includes three core modules: robot dance arrangement, intelligent music generation and multi-device collaborative control. Citation [3] ([https://blog.csdn.net/weixin\\_42511080/article/details/115808426](https://blog.csdn.net/weixin_42511080/article/details/115808426)) [7]  
(<https://sghexport.shaobserver.com/html/baijiaohao/2023/07/00>  
(<https://www.zghy.org.cn/item/682320743792300032>) [12]  
(<https://www.bilibili.com/video/av970410235/>) [14] (<https://3g.china>

```
```python
# -*- coding: utf-8 -*-
# Music Ball Master Control System v3.0
import time
import multiprocessing
from pygame import mixer
from robot_dance_ai import DanceAI
from music_generator import MelodyComposer
from stage_control import LightSystem, ConductorAI

class ConcertSystem:
    def __init__(self):
        self.orchestra = orchestra_controller() # 30-person band control
        self.dance_robots = dance_swarm(16) # Dance robot cluster
        self.light_system = lightsystem() # Intelligent lighting system
        self.audience_sensor = audience_feedback() # audience emotion perception

    # Opening procedure
    def opening_ceremony(self):
        # Speech synthesis announcement
        self.tts_announce ("Dear guests, the artificial intelligence symphony dance begins now!" )

    # Symphony prelude generation
    opener = MelodyComposer(style="symphonic").generate_opening(
        duration=180,
        instruments=["violin", "piano", "trumpet"]
    )

    # Multi-device collaborative startup
    with multiprocessing.Pool(4) as pool:
        pool.apply_async(self.orchestra.play, (opener,))
        pool.apply_async(self.light_system.start_sequence)
        pool.apply_async(self.dance_robots.initialize_pose)
```

```
# Dance Music Playing Core Logic
def dance_session(self, music_list):
    conductor = ConductorAI(tempo_detect_mode="real-time")
    for piece in music_list:
        # Dynamically generate band score
        score = MelodyComposer().arrange(
            piece,
            orchestra_size=32,
            complexity=0.8
        )

        # Command robot action generation
        conductor.load_score(score)
        beat_pattern = conductor.analyze_beat()

        # Multithreaded execution
        processes = [
            multiprocessing.Process(target=self.orchestra.play, args=(score,)),
            multiprocessing.Process(target=self.dance_robots.perform,
            args=(beat_pattern, "waltz"))
```